Winning Space Race with Data Science

Arpit Damani

25th May, 2022

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies
    - Data Collection through API
    - Data Collection through Web Scraping
    - Data Wrangling
    - Exploratory Data Analysis with SQL
    - Exploratory Data Analysis with Data Visualization
    - Interactive Visual Analytics with Folium
    - Machine Learning Prediction
- Summary of all results
    - Exploratory Data Analysis results
    - Interactive Visual Analysis results
    - Machine Learning Analysis results

Introduction

- Project background and context

  Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  - What are the factors which determine if the rocket will land successfully?
  - The interaction amongst various attributes that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models.

Data Collection

- The data was collected using various procedures
  - Data collection was done using the get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turned it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and filled in missing values where necessary.
  - We also performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is https://github.com/arpitdamani/IBM-Data-Science-Capstone-Project/blob/main/Data%20Collection%20API%20Lab.ipynb

1. Get request for rocket launch data using API

```
spacex_url = "https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

2. Using json.normalize method to convert json result to a dataframe

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

3. We then performed data cleaning and filling in the missing values

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, mean)
```

```
data_falcon9
```

Data Collection - Scraping

- We applied web scraping to extract the Falcon 9 launch records with BeautifulSoup.
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is https://github.com/arpitdamani/IBM-Data-Science-Capstone-Project/blob/main/Data%20Collection%20with%20Web%20Scraping.ipynb

1. Apply the HTTP GET method to request the Falcon9 Launch HTML page

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
# use requests.get() method with the provided static_url
response = requests.get(static_url).text
```

2. Create a BeautifulSoup object from the HTML Response

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'html5lib')
```

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

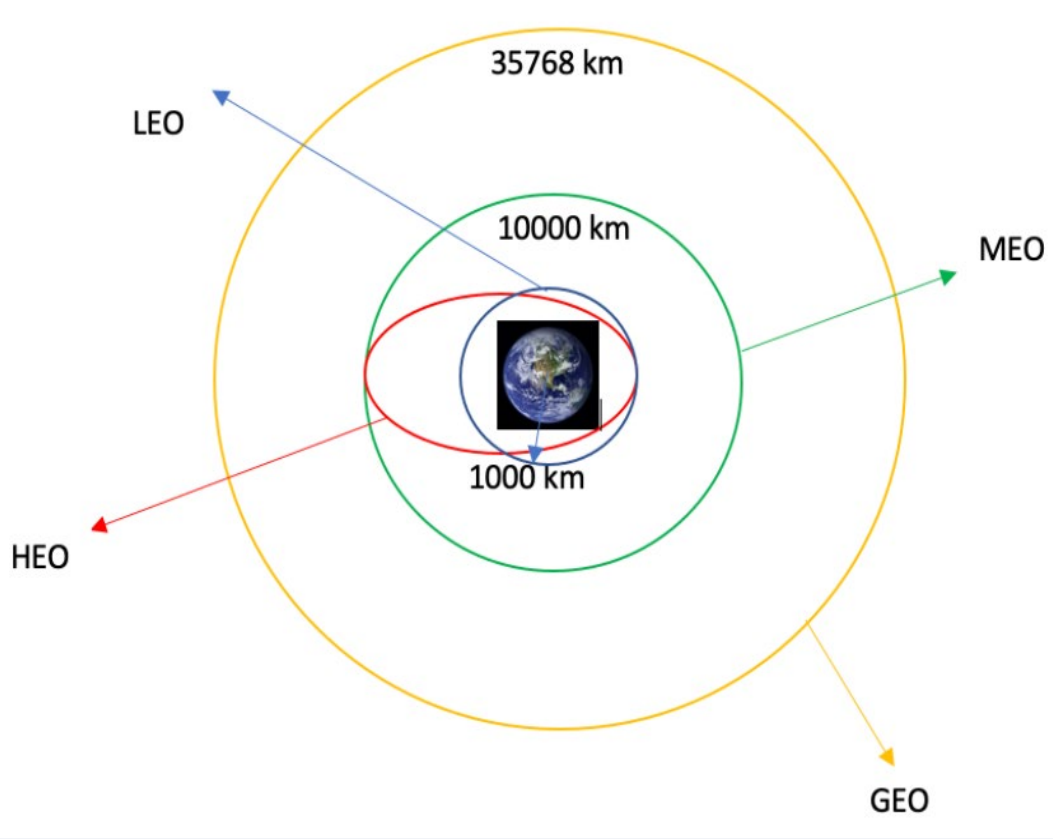
3. Extract all column/variable names from the HTML table header

```
column_names = []

for row in first_launch_table.find_all('th'):
    col_name = extract_column_from_header(row)
    if (col_name != None and len(col_name) > 0):
        column_names.append(col_name)
print(column_names)
```

4. Created a dataframe by parsing the launch HTML tables

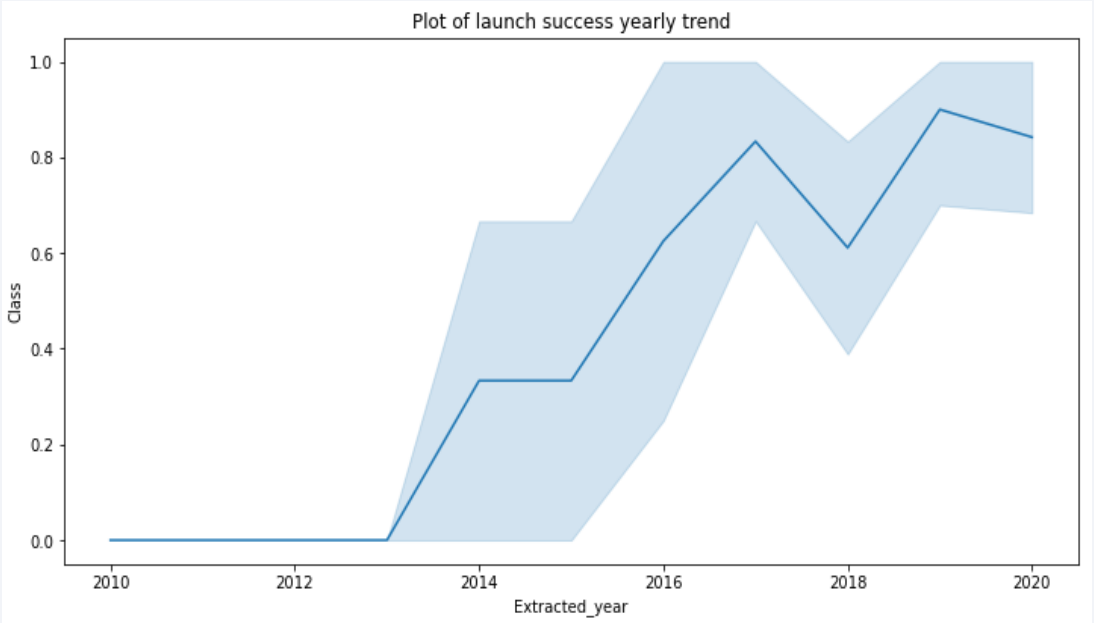
Data Wrangling

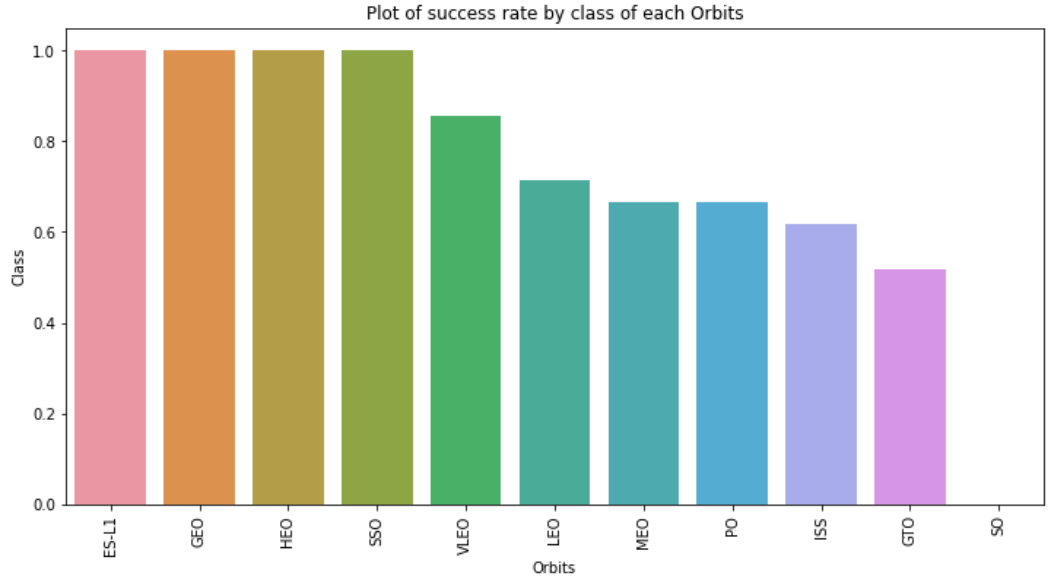


- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/arpitdamani/IBM-Data-Science-Capstone-Project/blob/main/EDA%20Lab.ipynb.

EDA with Data Visualization

- We explored the data by visualizing the relationship between Flight Number and Launch site, Payload and Launch site, Success rate of each orbit type, Flight Number and Orbit type, the Launch Success yearly trend.





- The link to the notebook is https://github.com/arpitdamani/IBM-Data-Science-Capstone-Project/blob/main/EDA%20with%20Visualization%20Lab.ipynb

EDA with SQL

- We loaded the SpaceX dataset into the DB2 database without leaving the Jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The date when the first successful landing outcome was achieved
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is https://github.com/arpitdamani/IBM-Data-Science-Capstone-Project/blob/main/EDA%20With%20SQL%20Lab.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are the launch sites near any highways, railways or coastlines?
  - If the launch sites are at a certain distance away from the cities?

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly Dash
- We plotted pie charts showing the total launches by a certain launch sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster versions.
- The link to the notebook is https://github.com/arpitdamani/IBM-Data-Science-Capstone-Project/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas library, transformed the data, and split our data into training and testing set.
- We built different machine learning models and tuned different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is https://github.com/arpitdamani/IBM-Data-Science-Capstone-Project/blob/main/Machine%20Learning%20Prediction%20Lab.ipynb

Results

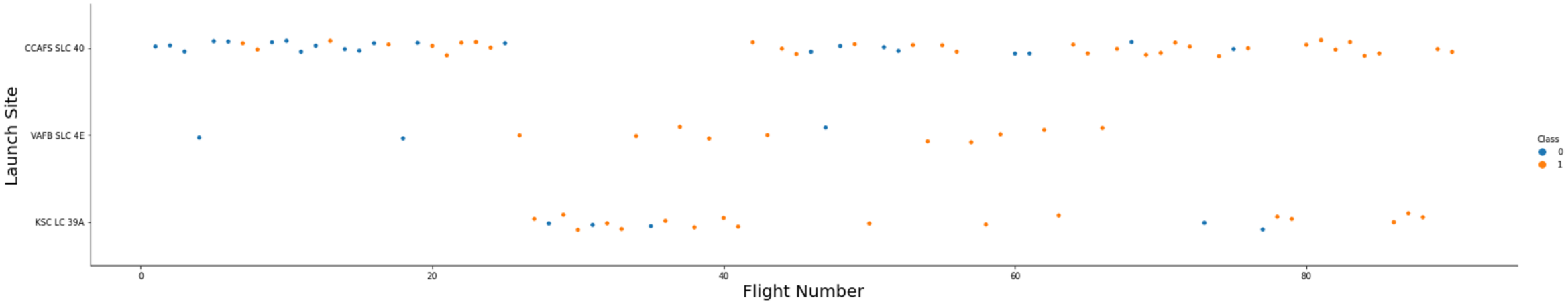
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

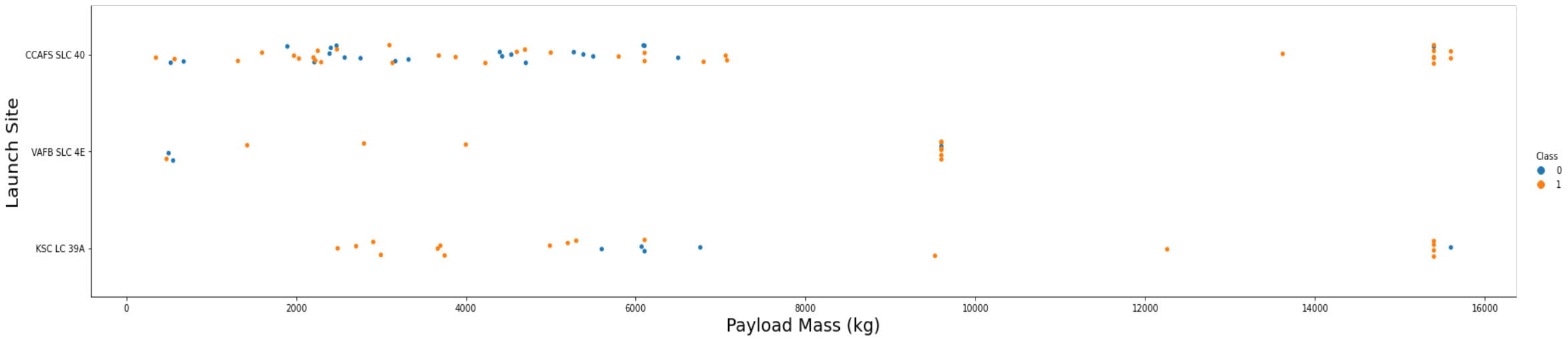
Flight Number vs. Launch Site

- From the plot, we found that there are more number of flights launched from the Florida launch sites as compared to the California launch site (VAFB SLC 4E).



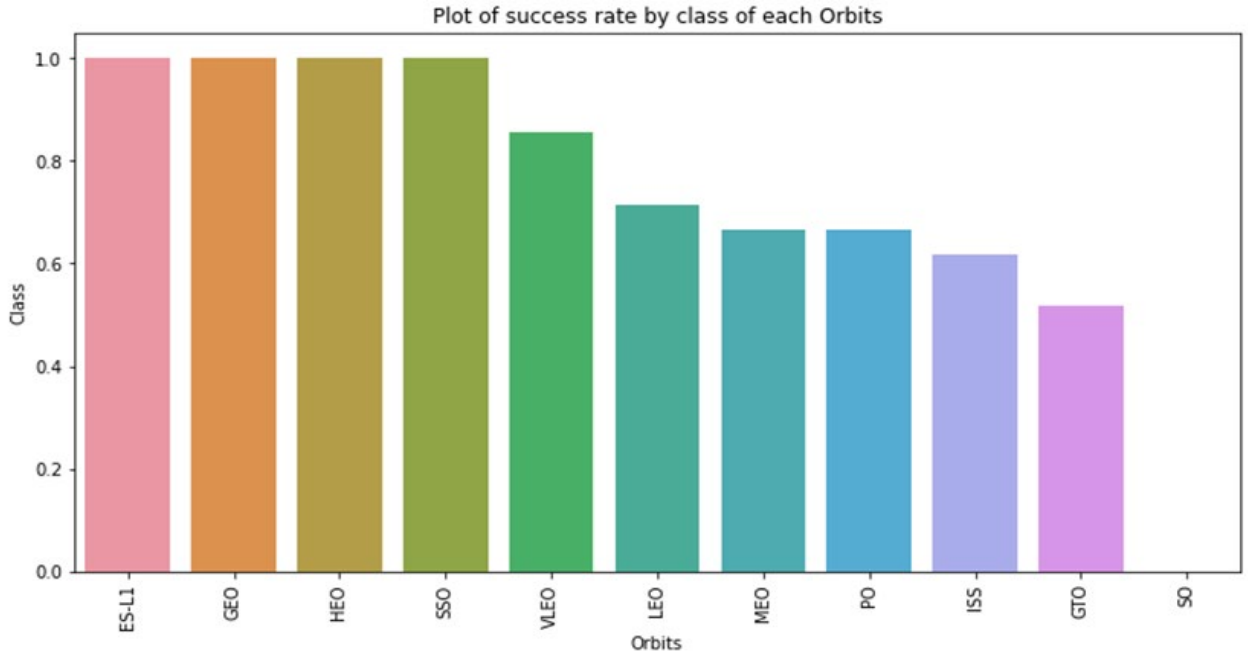
Payload vs. Launch Site

- From the plot, we found out that for VAFB-SLC launch site there are no rockets launched for heavy payload mass (greater than 10000kg).



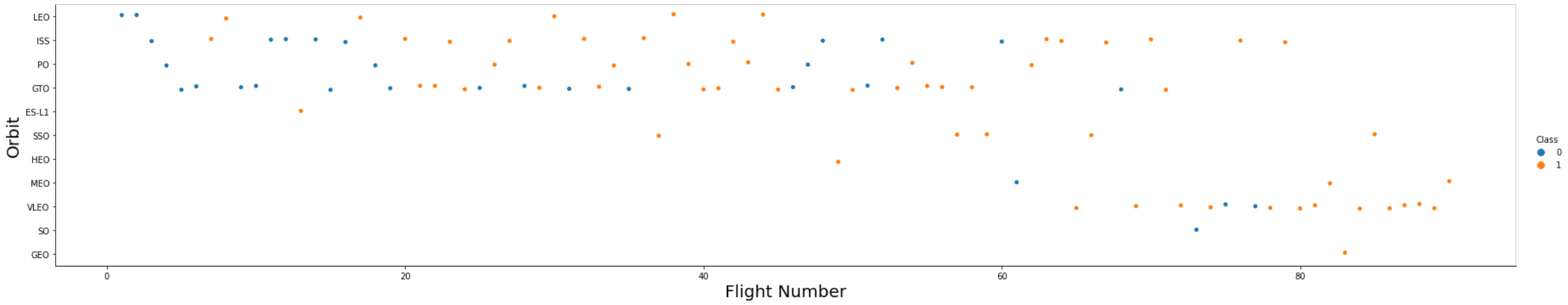
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



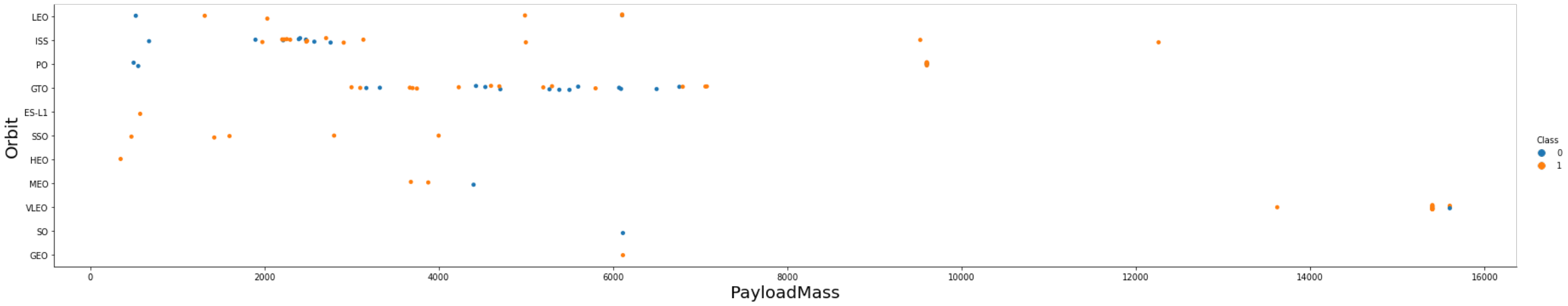
Flight Number vs. Orbit Type

- From the following plot we can observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there seem to be no relationship between the number of flights and the orbit.



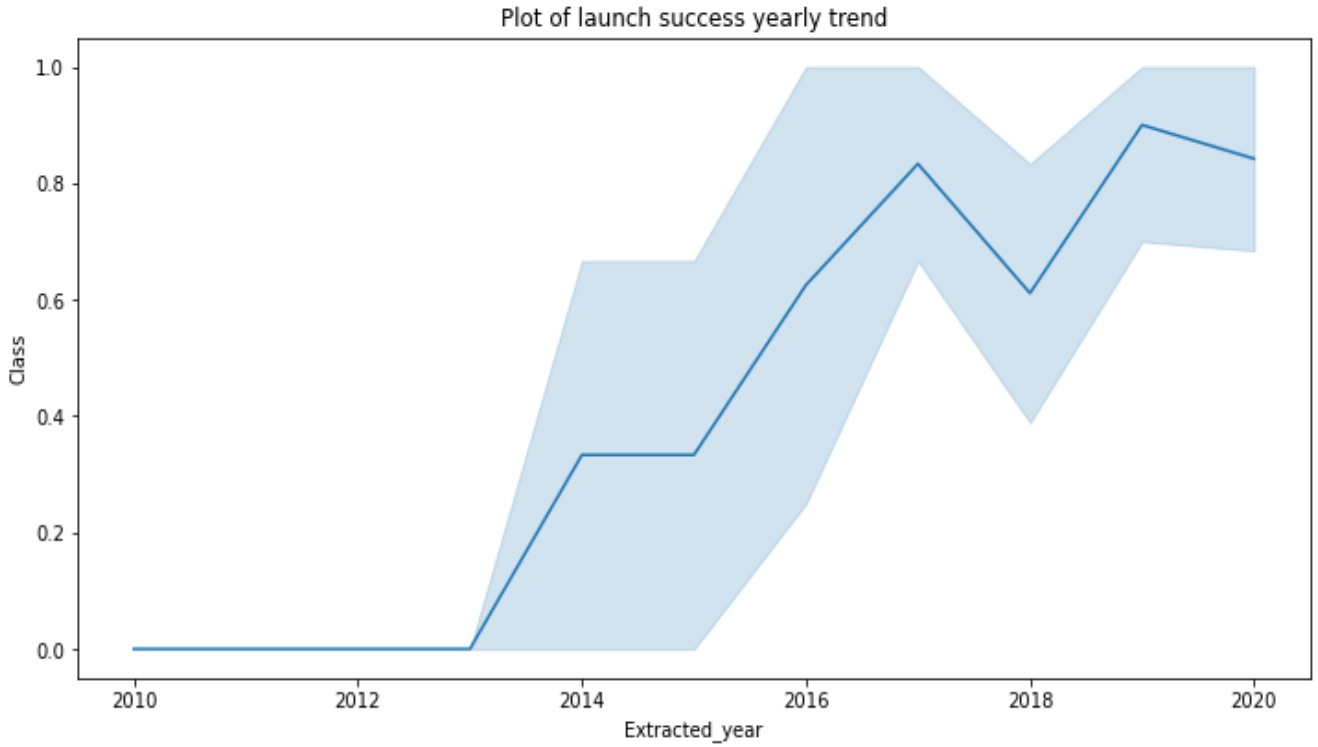
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing or positive landing rates are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

Display the names of the unique launch sites in the space mission

```
%sql SELECT DISTINCT(launch_site) from SPACEXTBL
```

```
 * ibm_db_sa://xpy88023:***@8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30120/bludb
Done.
```

launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Launch Site Names Begin with 'CCA'

Display 5 records where launch sites begin with the string 'CCA'

```
%sql SELECT * from SPACEXTBL WHERE launch_site LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://xpy88023:***@8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30120/bludb
Done.
```

DATE	time_utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing_outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- We used the ‘LIKE’ and ‘LIMIT’ operator to display 5 records where launch sites begin with ‘CCA’.

Total Payload Mass

- We calculated the total payload mass carried by boosters from NASA as ‘45596’ using the query below

Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql SELECT SUM(payload_mass__kg_) as total_payload_mass from SPACEXTBL WHERE customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://xpy88023:***@8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30120/bludb
Done.
```

total_payload_mass
45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as ‘2928’.

Display average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(payload_mass__kg_) as avg_payload_mass from SPACEXTBL WHERE booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://xpy88023:***@8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30120/bludb
Done.
```

avg_payload_mass
2928

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015.

List the date when the first successful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql SELECT min(date) as date from SPACEXTBL WHERE landing__outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://xpy88023:***@8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30120/bludb
Done.
```

DATE
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **BETWEEN** condition to determine successful landing with payload mass greater than 4000 but less than 6000.

```
%sql SELECT booster_version, payload_mass__kg_ from SPACEXTBL WHERE payload_mass__kg_ BETWEEN 4000 AND 6000 AND landing__outcome = 'Success (drone ship)'
```

```
 * ibm_db_sa://xpy88023:***@8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30120/bludb
Done.
```

booster_version	payload_mass__kg_
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

Total Number of Successful and Failure Mission Outcomes

- We used the GROUP BY clause on ‘mission_outcome’ to filter the total number of successful and failure mission outcomes.

List the total number of successful and failure mission outcomes

```
%sql SELECT mission_outcome, COUNT(mission_outcome) AS total_no_of_mission_outcomes from SPACEXTBL GROUP BY mission_outcome
```

```
 * ibm_db_sa://xpy88023:***@8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30120/bludb
Done.
```

mission_outcome	total_no_of_mission_outcomes
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql SELECT booster_version from SPACEXTBL WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) from SPACEXTBL)
```

```
 * ibm_db_sa://xpy88023:***@8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30120/bludb
Done.
```

booster_version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

- We used a combinations of the ‘WHERE’ clause, ‘LIKE’, and ‘AND’ conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015.

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%sql SELECT landing__outcome, booster_version, launch_site, DATE from SPACEXTBL WHERE DATE LIKE '2015%' AND landing__outcome = 'Failure (drone ship)'
```

```
 * ibm_db_sa://xpy88023:***@8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30120/bludb
Done.
```

landing__outcome	booster_version	launch_site	DATE
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-01-10
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015-04-14

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.
- We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

```
%sql SELECT landing__outcome, COUNT(landing__outcome) AS count from SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing__outcome ORDER BY count DESC
```

```
 * ibm_db_sa://xpy88023:***@8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30120/bludb
Done.
```

landing__outcome	COUNT
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

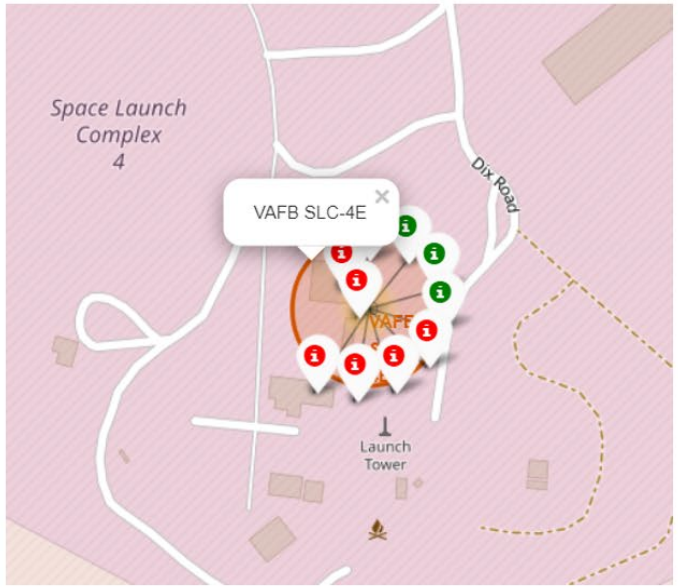
Section 3

Launch Sites Proximities Analysis

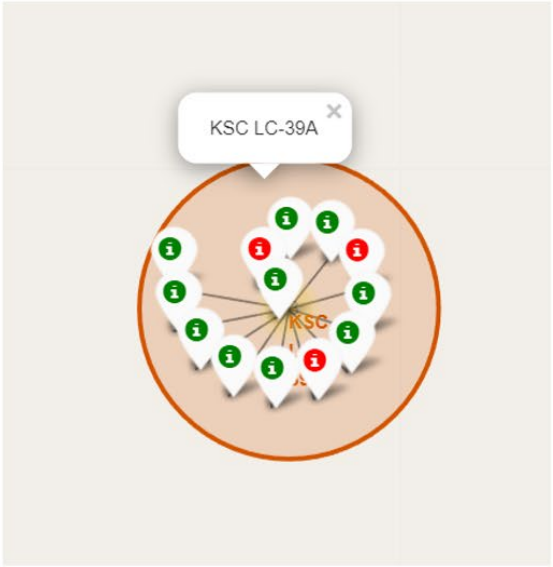
All global map markers launch sites' location

We can see that the SpaceX launch sites are located near the USA coasts – Florida & California

Markers showing launch sites with colour labels



California Launch Site

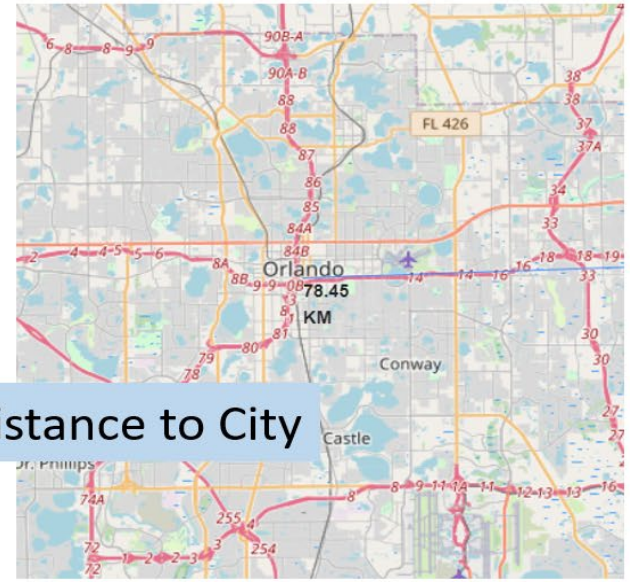


Florida Launch Sites

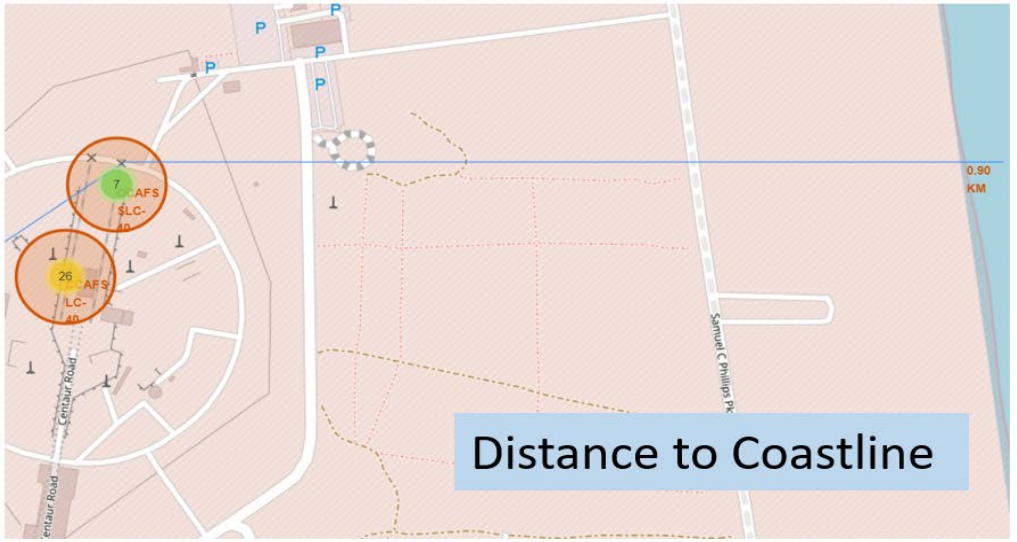
Green Marker - Successful Launches

Red Marker - Failed Launches

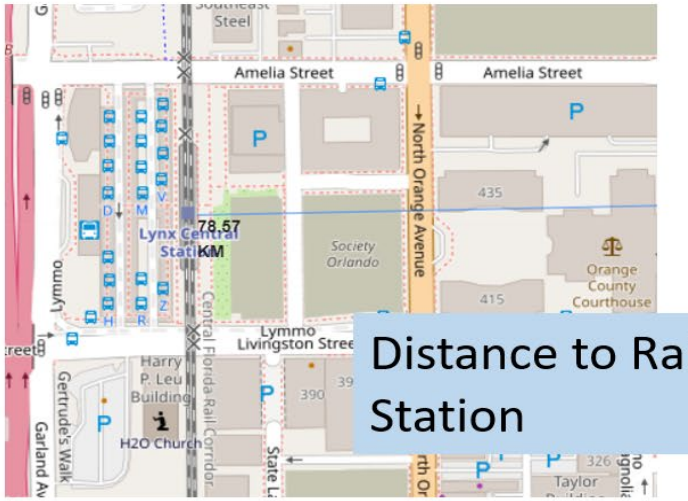
Launch sites' distance to its proximities



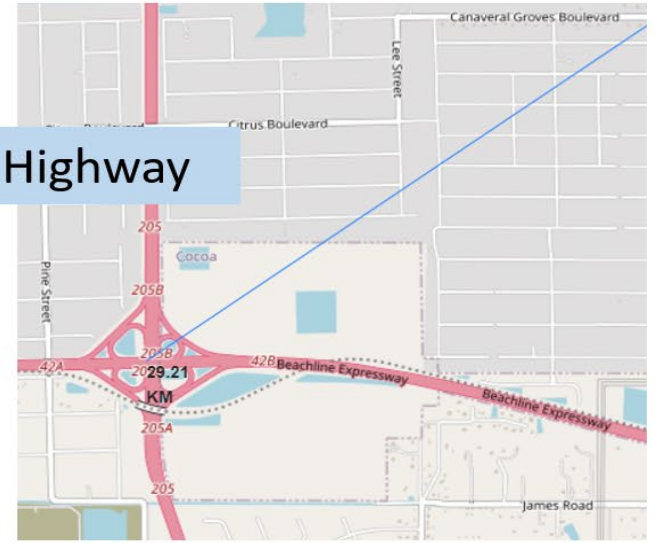
Distance to City



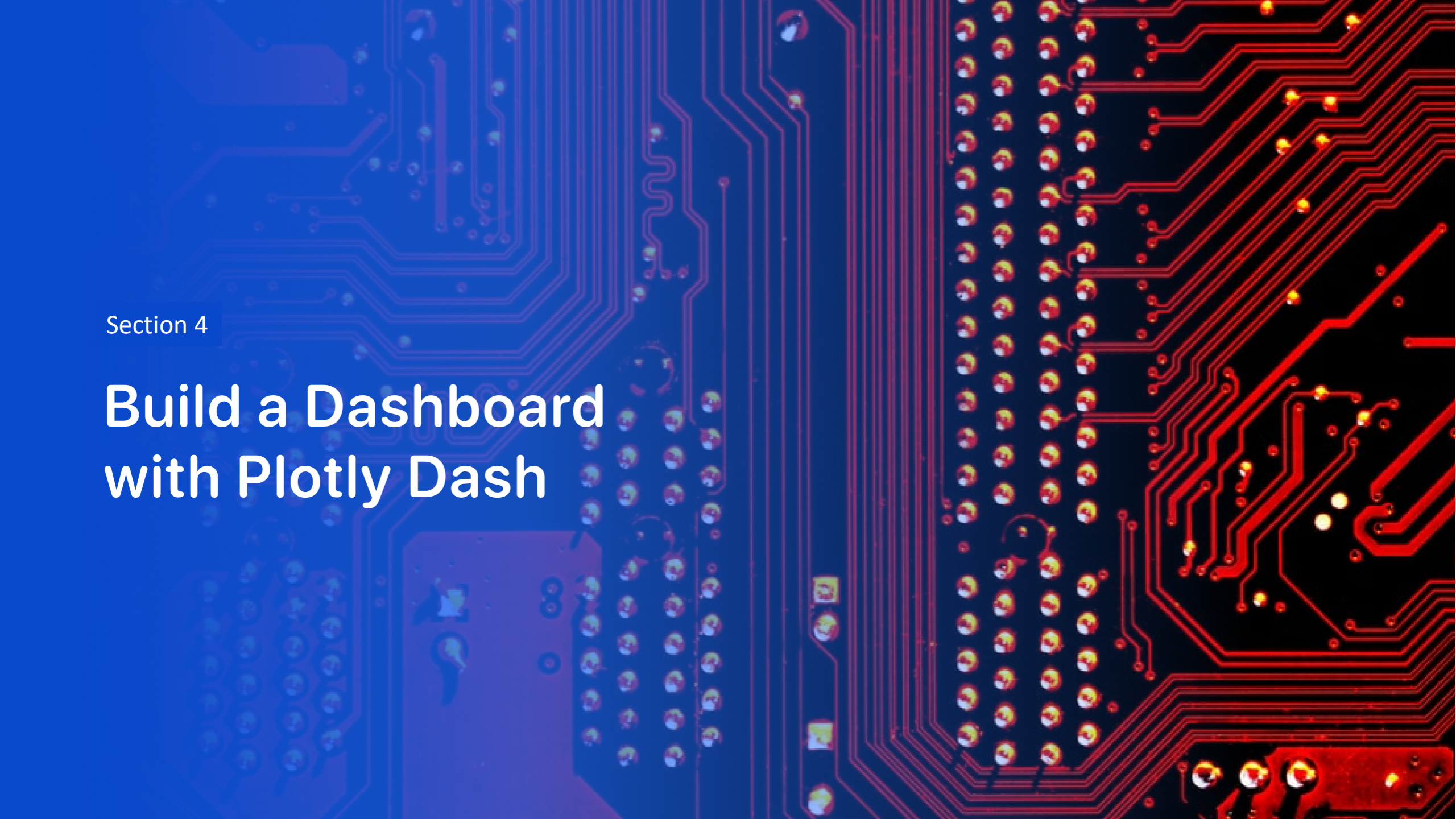
Distance to Coastline



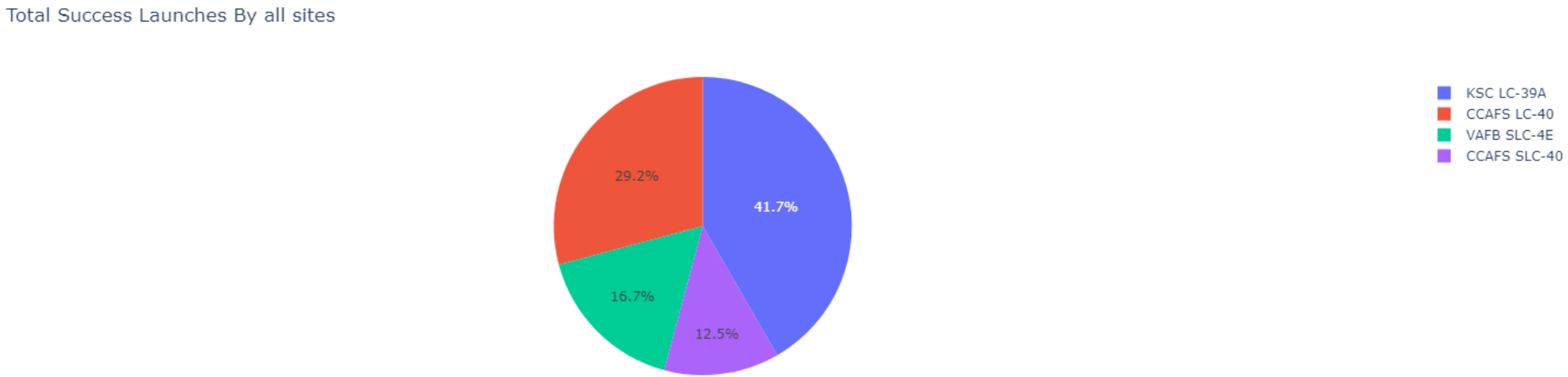
Distance to Railway Station



Distance to Highway

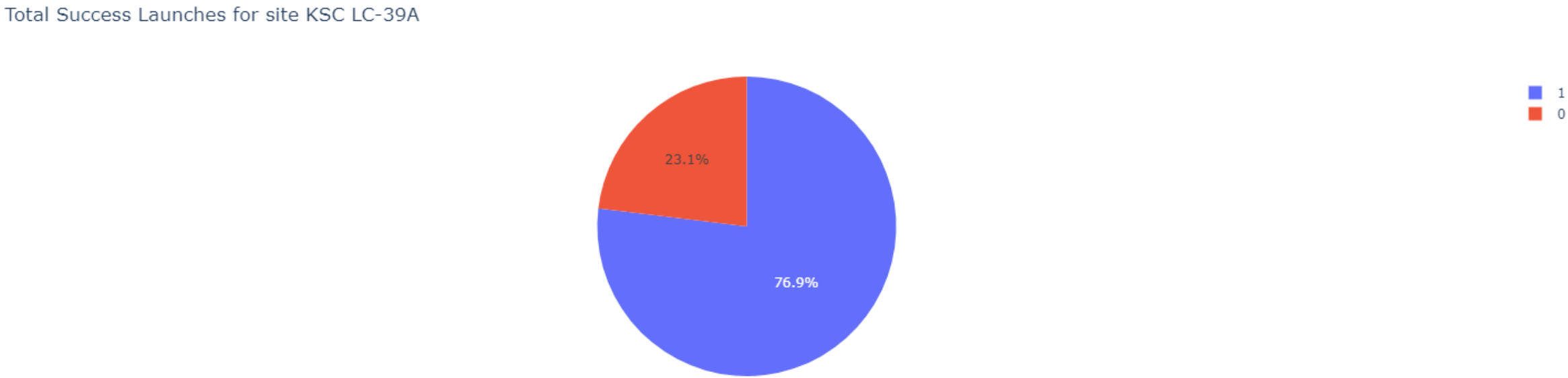

Section 4
Build a Dashboard with Plotly Dash

Pie chart showing success percentage by each launch site



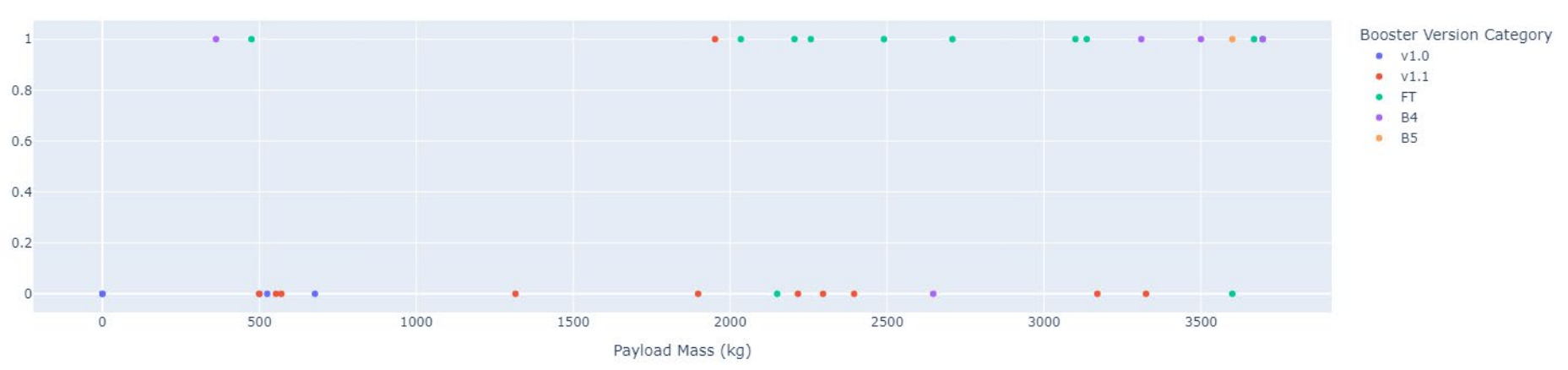
We can see that, the launch site KSC LC-39A had the most successful launches than all the other sites.

Pie chart for the launch site with the highest launch success ratio

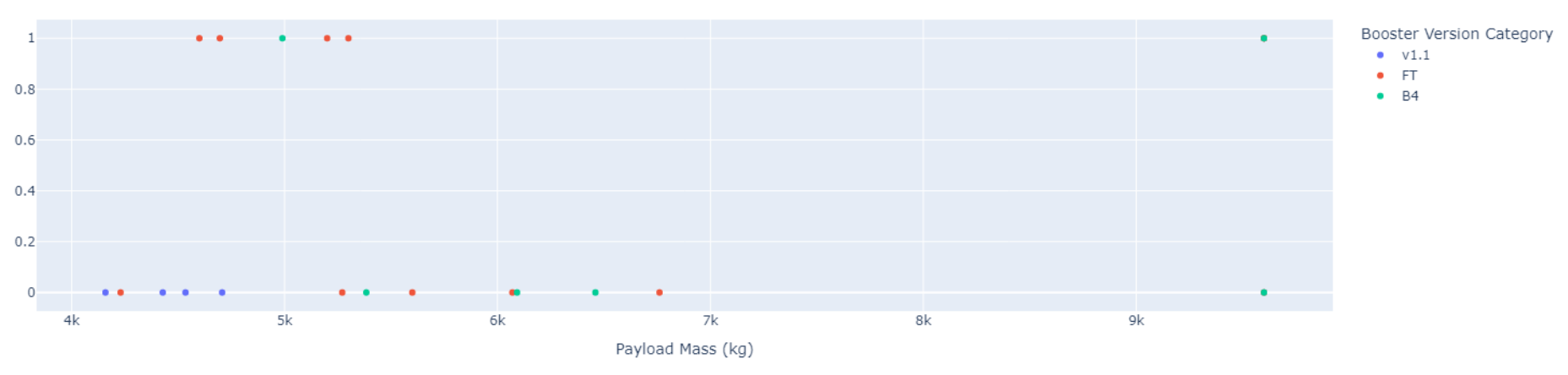


We can see that, the launch site KSC LC-39A has achieved the success rate of 76.9% whereas getting only 23.1% failure rate.

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



Low weighted payload 0kg - 4000kg



High weighted payload 4000kg - 10000kg

We can see that, the success rate for low weighted payloads is higher than the heavy weighted payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The Decision Tree classifier is the model with the highest classification accuracy.

```
models = {'KNeighbors':knn_cv.best_score_,
          'DecisionTree':tree_cv.best_score_,
          'LogisticRegression':logreg_cv.best_score_,
          'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- The Confusion Matrix for the Decision Tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.

Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision Tree classifier is the best machine learning algorithm for this task.

Thank you!